ONLINE FEATURE SELECTION FOR ACTIVITY RECOGNITION USING REINFORCEMENT LEARNING WITH MULTIPLE FEEDBACK

Taku Yamagata, Raúl Santos-Rodríguez, Ryan McConville, Atis Elsts

Department of Engineering Mathematics
University of Bristol
Bristol, UK

ABSTRACT

Recent advances in both machine learning and Internet-of-Things have attracted attention to automatic Activity Recognition, where users wear a device with sensors and their outputs are mapped to a predefined set of activities. However, few studies have considered the balance between wearable power consumption and activity recognition accuracy. This is particularly important when part of the computational load happens on the wearable device. In this paper, we present a new methodology to perform feature selection on the device based on Reinforcement Learning (RL) to find the optimum balance between power consumption and accuracy. To accelerate the learning speed, we extend the RL algorithm to address multiple sources of feedback, and use them to tailor the policy in conjunction with estimating the feedback accuracy. We evaluated our system on the SPHERE challenge dataset [1], a publicly available research dataset. The results show that our proposed method achieves a good trade-off between wearable power consumption and activity recognition accuracy.

Index Terms— Reinforcement Learning, Feature Selection, Activity Recognition, Embedded Systems

1. INTRODUCTION

There is significant interest in understanding activities of daily living (ADL) of people from a wide cross-section of society, but particularly within the healthcare domain. This is evidenced by the vast amount of studies undertaken utilizing accelerometers [2, 3], perhaps the most commonly used device for detecting ADL.

In the smart home context [1], low powered wearables constantly transmit their raw data to more computationally powerful devices where the actual processing is carried out. However, recently these wearables contain increasingly powerful microcontrollers that are capable of carrying out significant computation. Further, the energy trade-off between on-device computation and transmission is increasingly favouring on-device computation. In fact, when these wearables can adapt to context and make decisions online, energy savings can be made by, for example, dynamically reducing the sample rate of energy expensive wearable heart rate sensors [4]. In a similar vein, Elsts *et al.* [5] proposed moving a significant step in the activity recognition pipeline to the wearable, demonstrating the significant energy savings that can be made with on-board feature extraction.

However, one limitation of this work is that the set of features to be extracted are determined a priori. Intuitively, one can expect that the best set of features, in terms of the energy/accuracy trade-off, to be extracted to accurately recognize a specific activity depends on the context.

In this work, we propose a Reinforcement Learning (RL) based feature selection approach with the agent running on the wearable. To accelerate the RL algorithm learning, we also introduced a feedback mechanism from the smart home host processor, which has much more processing power and access to extra sensors. The feedback is incorporated into the RL algorithm based on the **Advise** algorithm [6] with two extensions - supporting multiple feedback sources and estimating their reliabilities of feedback. These extensions are useful, as the host processor could generate multiple feedback based on the extra sensors, and it is not clear the reliability of each of the feedback source. Our main contributions in this work are as follows.

- We present a methodology for energy efficient online selection of features from wearables based on context.
- We propose a novel RL learning algorithm that extends the work on [6] by supporting multiple feedback sources and estimating the reliability of feedback in an online fashion.

2. RELATED WORK

There has been much work in the general area of activity recognition [7], particularly wrist-based accelerometers, due to their performance and acceptability to users [8]. However, with the growing popularity of smart homes, potential arose

for these wearables to integrate with other devices within the home to benefit from increased context. In these settings, features are typically extracted from the raw acceleration data, after data collection, on much more powerful computers than the wearable that collected the data.

However, the transmission energy cost of the raw data is is expensive, and recent work [5] demonstrated that major energy savings can be made by moving the feature extraction to the wearable device, and transmitting features rather than raw data. However, in that work, features are used individually, and are chosen a priori. Our proposed method operates on groups of features, which can change dynamically based on context from other sensors.

The idea of utilizing RL to select features based on cost has been studied previously. Janisch *et al.* [9] pose the task of classification where each feature can be acquired for a cost, and the goal is to optimize the trade-off between the features' costs and classification performance. Similarly, Possas *et al.* [10] utilizes Deep RL to learn a policy to select between two activity recognition methods; one is the motion predictor, using a Long Short-Term Memory (LSTM) network, and the other is a vision predictor using both a Convolutional Neural Network and LSTM. While conceptually similar to our proposed approach, due to their deep nature, they both use considerably more power, and thus unsuitable for the setting of very low-power wearables. Our approach uses simplified discrete states to replace the deep network and employs feedback from other sensors to accelerate the agent's learning speed.

3. LEARNING ALGORITHMS

The goal of the our proposed algorithm is to learn a feature selection policy which achieves low power consumption with a low error rate. We contextualize the method within a smart home setting similar to the SPHERE platform [1], which consists of numerous different sensor modalities, including PIR and RGB-D cameras, as well as a wrist-worn accelerometer. Fig. 1 shows the overall structure of our AR system. Our wearable extracts features from sensor outputs and transmits the features to the host, which produces predictions of activities. To simplify the setting, the wearable has access to two groups of features, one group of low-power features and one group of complex, higher-power features. These are chosen based on their power consumption [5]. From these, the agent chooses the appropriate set of features to be computed before transmitting them to the host. Thus, the aim is to learn the best feature selection policy which achieves a good trade-off between power consumption and accuracy.

Our method compensates for the lack of computational resources on the device by taking into account feedback from other sensors for each agents action. This feedback is then used to shape the agents policy. Due to the role of each sensor, we will refer to them as *trainers*. This procedure aids finding the optimum policy and accelerates the agents learning. Furthermore, we extend the existing literature to incorporate multiple trainers and handle unreliable feedback. In our proposed method and evaluation, the feedback consists of passive infrared sensors (PIR) and RGB-D sensors that are placed throughout a residential home. The host processes this data, producing feedback that is transmitted to the wearable agent.

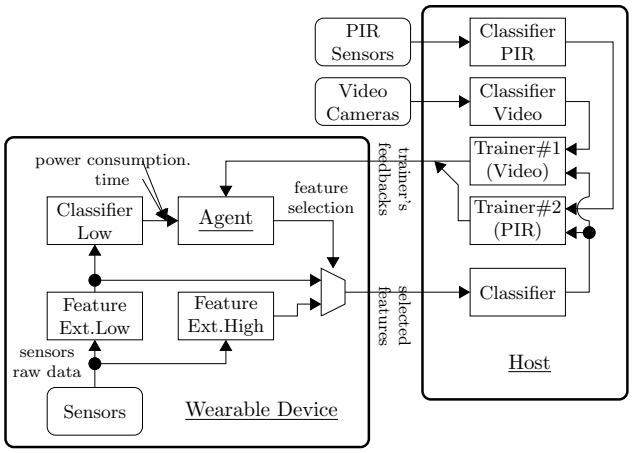

Fig. 1. Overall structure of the AR system.

3.1. RL with feedback

The reinforcement learning framework has two components, an *agent* and an *environment* [11]. The *agent* decides which action to take, and the *environment* reacts to the action and presents a new state to the *agent*. The *environment* also generates rewards, which are special numerical values whose sum the *agent* tries to maximize over time.

The interactions between the *agent* and the *environment* happen in discrete time steps, $t = 0, 1, 2, 3...$, and at each time the *agent* observes the state of the *environment* (s_t) and the reward (r_t) and then decides which action (a_t) to take next. The goal of the reinforcement learning algorithm is to learn a mapping from states to actions that maximizes the rewards over time through these interactions with the *environment*. The mapping is called the agent's *policy*, denoted by $\pi(s, a)$, which indicates the probability of choosing action a in state s.

3.1.1. Advise

In order to incorporate feedback into the RL algorithm we build upon the **Advise** algorithm [6] and thus provide a brief introduction to the approach. **Advise** assumes a binary feedback from a trainer that returns either 'right' or 'wrong' for a particular agent's choice of action. The feedback is accumulated for each state-action pair separately, and it is used to derive a trainer's policy, denoted by $\pi_F(s, a)$, which is

then used to modify agent's policy. Additionally, C is defined as the probability that the trainer gives the right (consistent) feedback, and assuming a binominal distribution, the trainer's policy is as follows.

$$\pi_F(s,a) = \frac{C^{\Delta(s,a)}}{C^{\Delta(s,a)} + (1-C)^{\Delta(s,a)}} \tag{1}$$

where $\Delta(s,a)$ is the difference between the number of positive and negative feedback from the trainer. The policy of the trainer is combined with $\pi_R(s,a)$ (policy from the underlying RL algorithm) by multiplying them together, so that the final policy becomes as

$$\pi(s,a) \propto \pi_F(s,a) \times \pi_R(s,a). \tag{2}$$

3.2. Extensions to Advise

While the **Advise** algorithm provides a mechanism for incorporating feedback, it has two major limitations. The first is that it requires the consistency level (C) prior to receiving feedback, which may be unknown or difficult to estimate in many applications. The second limitation is the restriction to a single trainer. When multiple trainers are available it may be beneficial to incorporate feedback from all sources. Further, by estimating their reliability, reliable trainer feedback may be incorporated while avoiding adversarial effects from unreliable trainers. Thus, in this work we extend **Advise** to take multiple trainers feedback into account while also treating the consistency level as an unknown parameter, estimating it in an online fashion.

3.2.1. Consistency Level Estimation

In this section we describe how to estimate the n_{th} trainer's consistency level ($C_{[n]}$). The estimation has two steps, the first step produces an estimate of the consistency level for a given state and action pair for the n_{th} trainer ($\mathcal{C}_{[n]s,a}$), and the second takes an average to obtain an universal $C_{[n]}$ for all state action pairs. In this subsection, all discussions are regarding a single n_{th} trainer, hence we omit the trainer's index $[n]$ from all the variables for simplicity. We will reintroduce the index in the next subsection.

First, we consider estimating the trainer's consistency level for a given state action pair ($\mathcal{C}_{s,a}$). With a given number of positive feedback ($h^+_{s,a}$) and negative feedback ($h^-_{s,a}$) on a given state action pair, this can be derived by maximizing the following log-likelihood function,

$$l(\mathcal{C}_{s,a}) = \log\left(p\left(h^+_{s,a}, h^-_{s,a}; \mathcal{C}_{s,a}\right)\right). \tag{3}$$

As there is no model to compute this likelihood function, we introduce a hidden parameter $\mathcal{O}_{s,a}$, and then marginalize out the hidden parameter to obtain the original likelihood function. The hidden parameter is a boolean that is 1 when a is the optimal action at state s, and 0 when it is not. Eq. 3 can be rewritten as:

$$l(\mathcal{C}_{s,a}) = \log\left(\sum_{\mathcal{O}_{s,a}} p\left(h^+_{s,a}, h^-_{s,a}, \mathcal{O}_{s,a}; \mathcal{C}_{s,a}\right)\right). \tag{4}$$

We then use the Expectation Maximization (EM) algorithm [12] to compute a maximum likelihood estimate of the consistency level ($\mathcal{C}_{s,a}$). The i_{th} iteration of the M-step can be written as follows,

$$\mathcal{C}^{(i+1)}_{s,a} = \frac{P_1 \cdot h^+_{s,a} + P_0 \cdot h^-_{s,a}}{h^+_{s,a} + h^-_{s,a}} \tag{5}$$

where $\mathcal{C}^{(i+1)}_{s,a}$ is the estimated consistency level at the i_{th} iteration, P_0 and P_1 are given as follows:

$$\begin{aligned} P_0 &= p\left(\mathcal{O}_{s,a} = 0 | h^+_{s,a}, h^-_{s,a}; \mathcal{C}^{(i)}_{s,a}\right). \\ P_1 &= p\left(\mathcal{O}_{s,a} = 1 | h^+_{s,a}, h^-_{s,a}; \mathcal{C}^{(i)}_{s,a}\right). \end{aligned} \tag{6}$$

The E-step fundamentally requires computing Eq. 6, using Eq. 1 and the probabilities derived from interaction with the environment. $P^Q_1(s,a)$ and $P^Q_0(s,a)$ are the probabilities of the optimal and non optimal action. As a result, they can be written as follows.

$$\begin{aligned} P_0 &= \frac{P^Q_0(s,a) \cdot \left(1 - \mathcal{C}^{(i)}_{s,a}\right)^{\Delta(s,a)}}{P^Q_1(s,a) \cdot \left(\mathcal{C}^{(i)}_{s,a}\right)^{\Delta(s,a)} + P^Q_0(s,a) \cdot \left(1 - \mathcal{C}^{(i)}_{s,a}\right)^{\Delta(s,a)}}. \\ P_1 &= \frac{P^Q_1(s,a) \cdot \left(\mathcal{C}^{(i)}_{s,a}\right)^{\Delta(s,a)}}{P^Q_1(s,a) \cdot \left(\mathcal{C}^{(i)}_{s,a}\right)^{\Delta(s,a)} + P^Q_0(s,a) \cdot \left(1 - \mathcal{C}^{(i)}_{s,a}\right)^{\Delta(s,a)}}. \end{aligned} \tag{7}$$

We set $P^Q_1(s,a) = \pi_R(s,a)$ and $P^Q_0(s,a) = 1 - \pi_R(s,a)$. The algorithm is summarized in Algorithm 1.

Now that we have derived an algorithm to estimate consistency level for each state-action by using the EM algorithm, we are going to summarize it over the state-action space to come up with one consistency level value for each trainer. In order to compute the consistency level, we run the recursive averaging method shown in Eq 8 for every state-action pair actually experienced by the agent.

$$C = C + \alpha \cdot (\mathcal{C}_{s,a} - C) \tag{8}$$

where $\alpha \in [0,1]$ is the learning rate, C is averaged consistency level and $\mathcal{C}_{s,a}$ is the estimated consistency level for the current state-action pair.

Additionally, we consider an approach which adaptively changes the learning rate based on the ratio of accuracy of the estimated $\mathcal{C}_{s,a}$ and the averaged C as follows,

$$\alpha = \alpha_0 \cdot \frac{\text{Accuracy of } \mathcal{C}_{s,a}}{\text{Accuracy of } C} \tag{9}$$

Algorithm 1 Consistency Level Estimation

Require: $P_0^Q(s,a)$, $P_1^Q(s,a)$, $h_{s,a}^+$ and $h_{s,a}^-$

$\Delta(s,a) \leftarrow h_{s,a}^+ - h_{s,a}^-$
$i \leftarrow 1$
$C^{(i)} \leftarrow 0.5$
while TRUE **do**
  $P_0 \leftarrow \frac{P_0^Q(s,a)\cdot\left(1-C^{(i)}\right)^{\Delta(s,a)}}{P_1^Q(s,a)\cdot\left(C^{(i)}\right)^{\Delta(s,a)}+P_0^Q(s,a)\cdot\left(1-C^{(i)}\right)^{\Delta(s,a)}}$
  $P_1 \leftarrow \frac{P_1^Q(s,a)\cdot\left(C^{(i)}\right)^{\Delta(s,a)}}{P_1^Q(s,a)\cdot\left(C^{(i)}\right)^{\Delta(s,a)}+P_0^Q(s,a)\cdot\left(1-C^{(i)}\right)^{\Delta(s,a)}}$
  $C^{(i+1)} \leftarrow \frac{P_1\cdot h_{s,a}^+ + P_0\cdot h_{s,a}^-}{h_{s,a}^+ + h_{s,a}^-}$
  if $C^{(i+1)} == C^{(i)}$ **then**
    break
  end if
  $i \leftarrow i+1$
end while
return $C^{(i)}$

where α_0 is a **base learning rate**, which is fixed and scaled by the ratio of the accuracies. The consistency level estimation uses two sources, information from the underlying RL algorithm (P_1^Q and P_0^Q) and the trainer's feedback ($h_{s,a}^+$ and $h_{s,a}^-$). Thus, we estimate these accuracies separately and combine them by multiplying them together to get the accuracy of the consistency level estimation. For the underlying reinforcement learning accuracy, we use as a metric the absolute value of state-action value function (Q function), added up over all actions.

$$\mathcal{Q}(s) = \sum_{a \in A} |Q(s,a)|. \tag{10}$$

For the trainer's feedback accuracy, we simply use the amount of feedback or the given state, i.e.,

$$\mathcal{H}(s) = \sum_{a \in A} h_{s,a}^+ + h_{s,a}^-. \tag{11}$$

The above metrics are used for estimating the accuracy of $\mathcal{C}_{s,a}$, and we use the following recursive averaging update to track these metrics for the averaged consistency level C.

$$\begin{aligned} \tilde{\mathcal{Q}} &= \tilde{\mathcal{Q}} + \alpha(\mathcal{Q}(s) - \tilde{\mathcal{Q}}). \\ \tilde{\mathcal{H}} &= \tilde{\mathcal{H}} + \alpha(\mathcal{H}(s) - \tilde{\mathcal{H}}). \end{aligned} \tag{12}$$

Then, we calculate the learning rate α by using $\mathcal{Q}(s)$, $\mathcal{H}(s)$, $\tilde{\mathcal{Q}}$ and $\tilde{\mathcal{H}}$.

$$\alpha = \alpha_0 \cdot \frac{\mathcal{Q}(s)\cdot\mathcal{H}(s)}{\tilde{\mathcal{Q}}\cdot\tilde{\mathcal{H}}}. \tag{13}$$

As $\alpha \in [0,1]$, we limit the upper value of α to be 1.0 by simply taking $\min(\alpha, 1.0)$. For our evaluation we fix the base learning rate (α_0) to $1.0/16.0$ as in practice it represents a good trade-off between the overall learning speed and suppressing noise in the consistency level estimation.

Algorithm 2 Consistency Level Estimation with an Adaptive Learning Rate

Require: α_0, $Q(s,a)$, $h_{s,a}^+$ and $h_{s,a}^-$
Require: C, $\tilde{\mathcal{Q}}$ and $\tilde{\mathcal{H}}$ persistent variables (C initialized 0.5, $\tilde{\mathcal{Q}}$ and

$\mathcal{Q}(s) \leftarrow \sum_{a' \in A} |Q(s,a')|$
$\mathcal{H}(s) \leftarrow \sum_{a' \in A} h_{s,a'}^+ + h_{s,a'}^-$
$\alpha \leftarrow \frac{\mathcal{Q}(s)\cdot\mathcal{H}(s)}{\tilde{\mathcal{Q}}\cdot\tilde{\mathcal{H}}} \cdot \alpha_0$
$C \leftarrow C + \alpha \cdot (C(s,a) - C)$
$\tilde{\mathcal{Q}} = \tilde{\mathcal{Q}} + \alpha(\mathcal{Q}(s) - \tilde{\mathcal{Q}})$
$\tilde{\mathcal{H}} = \tilde{\mathcal{H}} + \alpha(\mathcal{H}(s) - \tilde{\mathcal{H}})$
return C

3.2.2. Multiple Trainers

In order to incorporate multiple trainers, we assume each trainer has a different consistency level, and the n_{th} trainer's consistency level is denoted by $C_{[n]}$. The Bayes optimal method to combine probabilities from (conditionally) independent sources is multiplying them together [13], hence the policy for overall multiple trainers $\pi_F(s,a)$ can be derived as follows by employing each trainer's policy given in Eq.1,

$$\pi_F(s,a) \propto \prod_{n=1}^{N} \left(C_{[n]}\right)^{\Delta_{[n]}(s,a)} \tag{14}$$

where N is the number of trainers, and $\Delta_{[n]}(s,a)$ is the difference between positive and negative feedback on the state s and action a from the n_{th} trainer.

4. EVALUATION

To evaluate our proposed approach we use the SPHERE challenge dataset [1]. This publicly available dataset consists of human annotated activity labels from a variety of different sensors, aligning with our proposed setting. Importantly for this work, it contains acceleration data from a wrist-worn wearable sampled at 20Hz ($\pm$ 4G) from ten different participants, each following the same script within a residential house. We use the complete set of annotated 20 activities, which includes ambulation activities (e.g., walking, jumping), posture activities (e.g., standing, sitting) and transition activities (e.g., sit to stand, turning). The dataset also contains data from RGB-D cameras, which were placed in multiple rooms of the home, as well as passive infrared (PIR) sensors, both of which will be used as feedback sources. We reprocess the data to assign activity labels at the one second granularity.

4.1. RL Setup

RL defines a class of algorithms for solving a Markov Decision Process (MDP). A MDP is defined by the tuple (S, A, T, R, γ) for the set of possible states $S = (t, \mathcal{P}, c_{low})$,

where t is the elapsed time in minutes, $\mathcal{P}$ is power consumption in mC and c_{low} is the low energy feature classifier output. Because t and $\mathcal{P}$ are rounded to the closest integer, our state space is discrete. A is the set of actions, either using low energy features or high energy features. The low energy feature set includes four time domain features, namely the mean, min and max, as well as the number of zero crossings, while the high energy features are the low energy features plus higher energy features such as quartiles, histograms and spectral features. T is the state transition function. Finally, R is a reward function that is all zeroes except for the last state in the episode. The final state reward is calculated based on the total power consumption and average error rate in the episode as Eq. 15.

$$r = -\lambda \cdot p_e - (\mathcal{P}/\mathcal{P}_{tgt})^2 \tag{15}$$

where p_e is the error rate, which is the percentage of activities misclassified ($1 - accuracy$), $\mathcal{P}$ is the power consumption, $\mathcal{P}_{tgt}$ is the desired power consumption and λ is a positive real number controlling trade off between error rate and power consumption. In our experiment we set $\mathcal{P}_{tgt} = 16.7$mC and $\lambda = 1.0$. 16.7mC is derived by assuming 10mAh battery charge is allocated for feature extraction over a 30 day period. The reward function quadratically penalizes the normalized power consumption. We define the problem as an episodic task where each task is 20 minutes long, with a time step of 5 seconds. For the sake of simplicity, the Agents use *Q-Learning* (QL) [14] and *Boltzmann exploration policy*. The hyper parameters are the discount factor $\gamma = 0.99$, the learning rate $\alpha = 0.1$ and the temperature parameter for the Boltzmann exploration $\tau = 0.1$.

The two trainers are implemented on the host to generate feedback for the RL agent on the wearable. Each trainer has a classifier from one of the additional sensors, namely PIR or RGB-D cameras. The feedback is generated by comparing the extra sensor classification and the selected feature classification. If the low energy feature set classification result is same as the extra sensor classification result, positive feedback is given for low energy feature set and negative feedback for high energy feature set. If the low energy feature classification result is not same as the extra sensor classification result, we check if the high feature classification result is the same as the extra sensor classification result. If they match, and the current power consumption is still less than the target power consumption, it generates positive feedback for high feature set. Otherwise it does not generate any feedback.

4.2. Results

We will compare our proposed method against three different approaches, including two baselines. The first baseline is the use of high or low power features chosen at random with equal probability (Random). The second baseline will only use low power features (Fixed Low). We will also compare our method that uses multiple trainers with online consistency level estimation (Multi-Trainers) with one using Q-Learning (QL), which is commonly used with RL, often as a baseline.

The learning curves for each different approach can be seen in Figure 2, showing the total reward obtained each episode. This figure shows that our proposed method learns quicker and reaches a higher asymptote compared to QL, while the random baseline maintains remains stagnant at around -155 and the low power features are around the level that QL reaches after 10,000 episodes. Figure 3 and Figure 4 show how each algorithm learns the trade-off between the power consumption and error rate, which is the percentage of activities misclassified ($1 - accuracy$.) For reference, the performance of a classifier which labelled activities randomly would have an error rate of 0.81. It is clear that the Fixed Low feature approach has consistently the highest error rate, with QL approaching a similar level after around 15,000 episodes. The two approaches consistently maintaining a low error rate are the Multi-Trainers and the random approach, with the random approach typically having a slightly lower error rate. However, Figure 4 shows that the random approach has a significantly higher power consumption than all other methods, including over twice the power consumption of our Multi-Trainers method. While the Fixed Low consistently has a lower power consumption, it has the highest error rate and thus does not achieve a good balance. QL eventually reduces the power consumption to a level lower than our method, however as this occurs the error rate increases. Therefore, in comparison to the others, our method maintains a consistently low power consumption, while maintaining a low error rate. Finally, the learning curve for the consistency level is plotted in Figure 5. It shows that, as the agent learns the environment, the agent perceives that the feedback is consistent with what it has learned, leading to to an increase in the consistency level. Interestingly, it also shows a slightly higher consistency level for the RGB-D sensor based trainer, which is expected as RGB-D sensors are thought to provide more information than PIR sensors.

5. CONCLUSION

In this work we proposed a method for online feature selection for sensor streams by leveraging feedback from multiple trainers (alternative sensors) while estimating their consistency. We evaluated our approach on a publicly available activity recognition dataset, where the task was to reduce the energy consumption of a wearable device containing the RL agent while maintaining a low error rate. The evaluation demonstrated that our proposed method was the only approach able to maintain the error rate while reducing power consumption, thus achieving our objective. The baselines failed to achieve the balance between the error rate and power consumption. The random policy achieved the lowest error rate by consuming the highest power, and the fixed low pol-

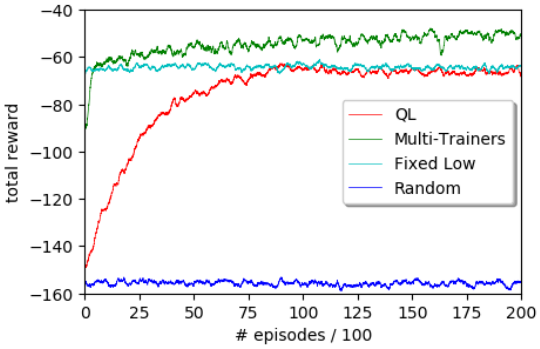

Fig. 2. Reward Learning Curves.

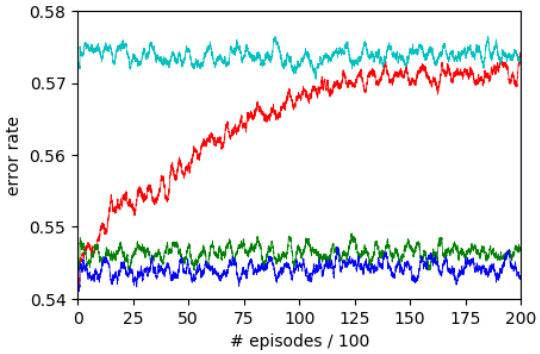

Fig. 3. Error Rate Learning Curves

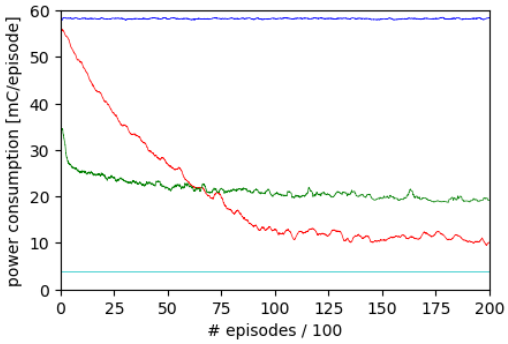

Fig. 4. Power Consumption Learning Curves

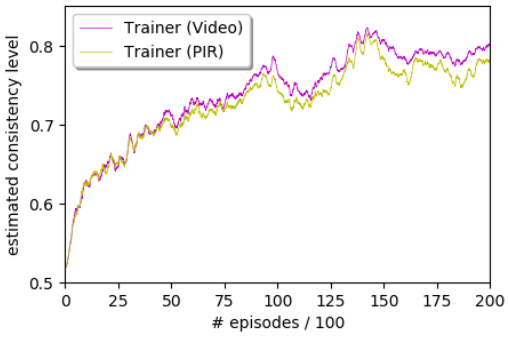

Fig. 5. Consistency Level Learning Curves

icy has the lowest power consumption by sacrificing the error rate. Q-Learning learned a policy in-between the previous two baselines, but fails to learn a policy to achieve the higher reward by reducing the error rate with small increments of its power consumption. We believe that this is because the error rate results have higher variance compared to the averaged error rate difference due to the feature selection, hence it is difficult to see the benefit of reducing the error rate by selecting the high energy feature set. Further, we found our motivation for learning the consistency level to be justified by the experiments. The RGB-D sensor, which would be expected to have a high consistency level, empirically was close to the PIR sensor. Thus, setting this consistency level a priori would not have been straightforward or optimal. We found that our method was robust to unreliable feedback and able to provide guidance to the agent to explore in the correct direction, ultimately achieving the best trade-off.

6. REFERENCES

[1] Niall Twomey, Tom Diethe, Meelis Kull, Hao Song, Massimo Camplani, Sion Hannuna, Xenofon Fafoutis, Ni Zhu, Pete Woznowski, and Peter Flach, "The sphere challenge: Activity recognition with multimodal sensor data," *arXiv preprint arXiv:1603.00797*, 2016.

[2] Aiden Doherty, Dan Jackson, Nils Hammerla, Thomas Plötz, Patrick Olivier, Malcolm H Granat, Tom White, Vincent T van Hees, Michael I Trenell, Christoper G Owen, et al., "Large scale population assessment of physical activity using wrist worn accelerometers: The uk biobank study," *PloS one*, vol. 12, no. 2, pp. e0169649, 2017.

[3] Ryan McConville, Raul Santos-Rodriguez, and Niall Twomey, "Person identification and discovery with wrist worn accelerometer data," in *ESANN*, 3 2018, pp. 615–620.

[4] Ryan McConville, Gareth Archer, Ian Craddock, Herman ter Horst, Robert Piechocki, James Pope, and Raul Santos-Rodriguez, "Online heart rate prediction using acceleration from a wrist worn wearable," in *KDD Workshop on Machine Learning for Medicine and Healthcare*, 8 2018.

[5] Atis Elsts, Ryan McConville, Xenofon Fafoutis, Niall Twomey, Robert Piechocki, Raul Santos-Rodriguez, and Ian Craddock, "On-board feature extraction from acceleration data for activity recognition," in *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2018, pp. 14–16.

[6] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz, "Policy shaping: Integrating human feedback with reinforcement

learning," in *Advances in neural information processing systems*, 2013, pp. 2625–2633.

[7] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, Third 2013.

[8] Niall Twomey, Tom Diethe, Xenofon Fafoutis, Atis Elsts, Ryan McConville, Peter Flach, and Ian Craddock, "A comprehensive study of activity recognition using accelerometers," *Informatics*, vol. 5, no. 2, 6 2018.

[9] Jaromr Janisch, Tom Pevn, and Viliam Lis, "Classification with costly features using deep reinforcement learning," in *AAAI*, 2019.

[10] Rafael Possas, Sheila Pinto Caceres, and Fabio Ramos, "Egocentric activity recognition on a budget," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5967–5976.

[11] Richard S. Sutton, *Reinforcement Learning*, The MIT Press, 1998.

[12] Arthur P Dempster, Nan M Laird, and Donald B Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[13] C Bailer-Jones and K Smith, "Combining probabilities," *Data Processing and Analysis Consortium (DPAS), GAIA-C8-TN-MPIA-CBJ-053*, 2011.

[14] Christopher John Cornish Hellaby Watkins, *Learning from delayed rewards*, Thesis, 1989.